AFRL-AFOSR-JP-TR-2016-0047

Omnidirectional learning -- A new paradigm for learning to predict any set of variables from any other set

Francois Petitjean
MONASH UNIVERSITY

05/06/2016
Final Report

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ IOA
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved

OMB No. 0704-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From - To)*
06-05-2016	Final	20 Apr 2015 to 19 Apr 2016

4. TITLE AND SUBTITLE

Omnidirectional learning -- A new paradigm for learning to predict any set of variables from any other set

5a. CONTRACT NUMBER
FA2386-15-1-4017

5b. GRANT NUMBER
Grant 15IOA017

5c. PROGRAM ELEMENT NUMBER
61102F

6. AUTHOR(S)

Francois Petitjean

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

MONASH UNIVERSITY
WELLINGTON RD
CLAYTON, 3800 AU

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

AOARD
UNIT 45002
APO AP 96338-5002

10. SPONSOR/MONITOR'S ACRONYM(S)

AFRL/AFOSR/IOA(AOARD)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

15IOA017

12. DISTRIBUTION/AVAILABILITY STATEMENT

Distribution Code A: Approved for public release, distribution is unlimited.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Awards: Best Paper Honorable Mention Award at the SIAM (Society for Industrial and Applied Mathematics Conference on Data Mining (SDM). Paper: Scaling log-linear analysis to datasets with thousands of variables. Businesses and Government are investing heavily in their data assets. As data quantities continue to grow rapidly, it is increasingly difficult to extract maximum value from those data stores. Learning to predict the future from past observations is one of the key components that make it possible to bring value to data.

To date, much of the research effort has been devoted to drawing predictions about a single pre-defined target variable, such as predicting the magnitude of the global warming, or the probability of developing cancer. However, in many real-world applications, what we wish to predict can change dramatically from one instance to the next, e.g. from one tactical situation to another or from one client to another. State-of-the-art techniques only provide ad-hoc solutions to this problem, because learning one model for every possibly encountered situation does not scale to big data assets. This project investigated methods for learning a single model that can effectively and efficiently predict all unobserved variables from the currently available evidence. New technologies were developed to learn models with this property from large and highdimensional data. The results show that the developed techniques offer a gain of 4 orders of magnitude in computation time over the state of the art.

15. SUBJECT TERMS

Data Mining

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			LUTZ, BRIAN
U	U	U	SAR	18	19b. TELEPHONE NUMBER *(Include area code)* +81-42-511-2000

Final Report for AOARD Grant FA2386-15-1-4017

Omnidirectional learning – A new paradigm for learning to predict any set of variables from any other set

29 April, 2016

Name of Principal Investigators (PI and Co-PIs): Dr François Petitjean
- e-mail address : francois.petitjean@monash.edu
- Institution : Monash University
- Mailing Address : Monash University, 25 Exhibition Walk, VIC 3800, Australia
- Phone : +61 3 990 59182
- Fax : +61 3 990 55159

Period of Performance: 04/30/2015 – 04/29/2016

Abstract:

Businesses and Government are investing heavily in their data assets. As data quantities continue to grow rapidly, it is increasingly difficult to extract maximum value from those data stores. Learning to predict the future from past observations is one of the key components that make it possible to bring value to data.

To date, much of the research effort has been devoted to drawing predictions about a single pre-defined target variable, such as predicting the magnitude of the global warming, or the probability of developing cancer. However, in many real-world applications, what we wish to predict can change dramatically from one instance to the next, e.g. from one tactical situation to another or from one client to another. State-of-the-art techniques only provide ad-hoc solutions to this problem, because learning one model for every possibly encountered situation does not scale to big data assets.

This project investigated methods for learning a single model that can effectively and efficiently predict all unobserved variables from the currently available evidence. We developed new technologies to learn models with this property from large and high-dimensional data. Our results show that our techniques offer a gain of 4 orders of magnitude in computation time over the state of the art.

1. Introduction:

Data analytics increasingly underpin many core processes in industry, commerce, governance and science. Data is only a key strategic asset because knowledge discovery methods make it valuable. As data quantity inexorably rises, more effective analytic techniques that can extract greater information from big data will add tremendous value. To date, much big data research has been devoted to drawing predictions about a single target variable: the magnitude of global warming, for example, or the probability of developing cancer. Classification is the task that aims

Name	Gender	Age	Weight	…	Cancer prediction
James	M	25-30	70kg-75kg	…	No
Johnson	F	75+	60kg-65kg	…	Yes
…					

Figure 1: classification example

at predicting the value taken by a categorical variable from observations over a pre-defined set of variables. In this very mature field of research, many algorithms have been proposed to learn how to draw accurate predictions from data [1] – see Figure 1 for an example.
However, classification is limited to predicting a *predefined* and *unique* variable.
This project sought to develop methods that learn how to effectively and efficiently predict *any set of variables* from any other set of variables. This is necessary because, in many real-world applications, what we know and what we wish to predict can change dramatically from one instance to the next, e.g. from one tactical situation to another or from one client to another. Traditional classification models can be of limited value to data practitioners because they fail to provide flexibility to predict whatever happens to be unobserved in any given context. Consider a bank adviser meeting a new client for the first time. From one new client to another, the bank adviser might be interested in predicting very different variables; from the likely income of the new client's household, to the likelihood of him or her to be interested in a specific insurance policy. Moreover, the information (variables) that the adviser might be able to collect from one new client to the next might be very different, depending on the course of the conversation. This process is exemplified in Figure 2 below, where the information collected by the adviser are depicted in blue, while the predicted values are depicted in green.

Name	Gender	Age	Income	partner	#children	…	#cars	Income protection	Education	#BR house	Exercise
James	M	25-30	$50k-$60k	No	None	…	1	No	College	1	Yes
Johnson	F	40-45	$150k+	Yes	2-3	…	3	Yes	College	5+	Yes
Smith	F	65-75	$50k-$60k	Yes	None	…	1	No	High-school	2-4	No
						…					

Figure 2: Example illustrating the need for more flexible learners - in blue: the information the adviser has acquired about his or her new client over the course of the conversation; in green the information predicted by an ideal system.

The same needs arise in many domains and situations. In systems supporting medical diagnosis, general practitioners might be interested in predicting different sets of variables for different patients while having different information about their medical condition and history. In asset management, financial advisers might wish to study the likelihood of different stocks gaining value after having learned about any of many different types of information about the rest of the market).
We call this task 'Omnidirectional learning': being able to learn from data described by a fixed set of variables Z, and being able to predict any subset of variables $Y \subset Z$, from any subset of the remaining ones $X \subseteq Z \backslash Y$.

Progress and limitations to date in this field of research.
'Omnidirectional learning' is not a standard term in the literature, and actually corresponds to a task that has only previously received limited attention and failed to be recognized as a distinct task requiring specialized methods. We now review the state of the research literature in the techniques that are related to this task, and explain how no existing algorithm can consistently fulfill the function of omnidirectional learner.
Over the last decades, there has been significant research interest in learning classification models from data. Classification learners aim at estimating the

probability of a target variable y (the class), given the values taken by a set X of M variables $\bar{x} = \{x_1, \cdots, x_M\}$; i.e. constructing a model of $p(y|\bar{x})$. Since the 1950s, a large variety of classification algorithms and strategies have been studied, with different properties, behaviors and cases they are particularly suitable: from Naïve Bayes, logistic regression [5] and decision trees, to SVMs [14] and Random Forest [15]. The reader can refer to [1] for a recent review of the main classification algorithms. A naïve way to address our omnidirectional task would thus be to utilize state-of-the-art classification algorithms, and compose a solution that learns to predict every possible variable from every possible subset of the remaining ones.

There is a first functional objection to the use of independent classifiers for omnidirectional learning: predicting every target variable of Y independently completely ignores the complex dependencies that might exist between them. For example, *number of children* and *number of bedrooms* are interdependent, and so predicting them separately will inevitably lead to important inaccuracies.

The second objection is functional: using independent classifiers for omnidirectional learning is impossible for most datasets, because this requires learning a different model for every possible target variable, i.e. 2^M models.[1] The only way to avoid having to build an exponential number of models would be to use models that can handle missing values. That would make it possible to build a single model per target variable and treat the other unobserved variables during classification as *unknown*. Three main strategies have been identified to deal with missing values at prediction time [2]:

1. *Discard instances*: discarding instances with missing values. This goes against the aim of omnidirectional learning, because all instances will have unobserved variables and hence be discarded.
2. *Imputation*: estimating the value or distribution of the unobserved variables, which would produce a typical chicken-and-egg problem for omnidirectional learning, because if variables X_1 and X_2 are unobserved, we would need to estimate X_2 to predict X_1 and vice versa.
3. *Reduced-feature Models*: using a different model, constructed to contain only the observed variables of test instance. There would then be three possible strategies:
    a. Learn all the possible models in advance, which we have shown above to be unfeasible for datasets with more than a dozen variables.
    b. Learn the whole model at classification time – a strategy known as "lazy classification" [3] – for which the time required to perform the classification would be incompatible with most applications.
    c. Marginalize over the unobserved variables of the learned model, which is only feasible for either simple models like Naïve Bayes – which will often not provide competitive predictions – or if only a few variables are unobserved.[2] That is contrary to the specifications of our omnidirectional framework.

[1] With the target variable fixed, each remaining variable has to be either *evidence/observed*, or *missing*, which leads to 2^M possible models.

[2] In the general case, marginalizing is exponential with the number of variables over which the marginalization is performed.

Thus, composing an omnidirectional learner from existing classifiers would be both unsound and unfeasible.

Graphical models [4] constitute a more consistent way to target omnidirectional learning, because:

1. They are models of the joint probability with no particular class, and can thus model complex dependencies between the variables.
2. They have efficient *inference* algorithms that make it possible to marginalize the joint probability over any set of variables. This would make it possible to compute the probability (or *belief*) over our set Y of *unobserved* variables, conditioned on some *evidence* over the remaining set of variables (X).

However, using graphical models for omnidirectional learning faces three main scientific obstacles:

Obstacle #1: How to efficiently marginalization over any set of variables? Echoing our discussion about the treatment of missing values for classification, making predictions from models of the joint probability requires marginalization. When a single variable z_1 is the target of the prediction and all values of the other variables are known, the conditional probability is obtained by marginalization over the target:

$$\hat{p}(z_1|z_2,\cdots,z_M) = \frac{\hat{p}(z_1,z_2,\cdots,z_M)}{\sum_{v\in Dom(z_1)}\hat{p}(v,z_2,\cdots,z_M)}$$

However, in the general case, such marginalization is only possible for one target variable; while omnidirectional learning requires being able to get a prediction about several (and potentially hundreds) of variables. As we have described above, this process is called *inference* or *belief propagation* for graphical models. It works by iteratively updating local probabilities depending on local neighborhoods of the graph, until convergence of the marginal probabilities. However, for general graphical models, only approximate algorithms exist (e.g. loopy belief propagation), for which convergence to actual marginals is uncertain [6].

Obstacle #2: How to learn from large and high-dimensional datasets? Learning graphical models from data has been of major interest since the 1990s with various methods proposed for log-linear models [5], Bayesian networks [7,8] and Markov Random Fields [9,10]. However, to the best of our knowledge, other than our own work in the area [12,13], no state-of-the-art method can efficiently learn from datasets with more than about 50 variables [3] without making strong assumptions about the distribution from which the data is drawn. This is, for example, the case for methods using l1-regularizers [9,25] which assume that every variable will interact with a very low number of variables, and for the PC/IC algorithms [26] which assume that the conditional independencies can be discovered with reduced subsets of variables. These assumptions lower the accuracy and reliability of the results.

In this project, we have focused on effectively and effectively **learn junction tree models**. Junction trees are a perfect class of graphical models for omnidirectional learning because they allow for efficient and exact marginalization over any set of variables, which directly solves Obstacle #1. This project focused on

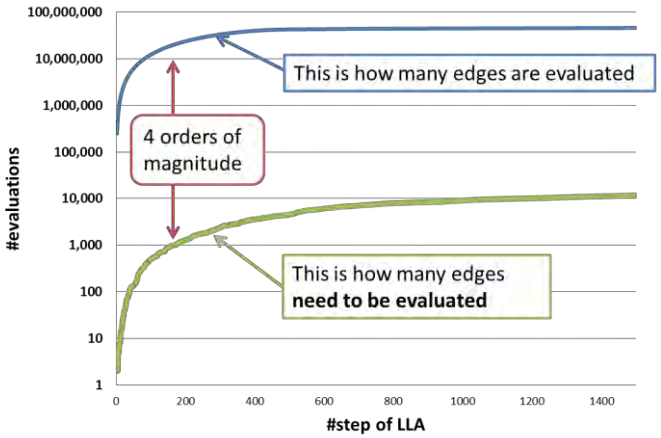
[3] See for example the running times in hours for most state-of-the-art methods on datasets with no more than 50 variables in [10].

the learning of the structure of junction tree models for data assets with 1,000+ variables.

2. Our approach: Prioritized Chordalysis

In a series of papers published in 2013 and 2014 [12,13], we had shown how to effectively learn junction tree models (also known as decomposable models) from data with medium dimensionality of up to 100 variables. However, further scalability remained limited, because state-of-the-art techniques have to examine every edge at every step of the search. For example, learning a junction-tree with the state-of-the-art for high-dimensional data requires more than 3 days for a dataset with 700 variables (see our experiments on the Protein dataset below).

The technology and algorithms developed in this project are based on the idea that, at every step of a forward search for the best graphical model, it is only necessary to reconsider a subset of edges for addition to the successively refined models. Let us motivate this idea with another real-world dataset representing 30,000 news articles described over 500 variables (see the description of dataset 'ABC News' in the experiments below for more details). On the one hand, we recorded how many edges are examined by the LLA process. We report this number over the course of the LLA process in the top curve in figure on the right: the process examines the addition of more than 10,000,000 edges. On the other hand, we looked at how many edges actually lead to the same evaluation of the model between successive steps of the search. We report the difference - i.e. the number of times that edges need to be re-examined after the very first step - in the bottom curve of the figure above: only about 10,000 edges' additions require re-examination. Note that the remainder of this report will make it clear how this graph can be generated.

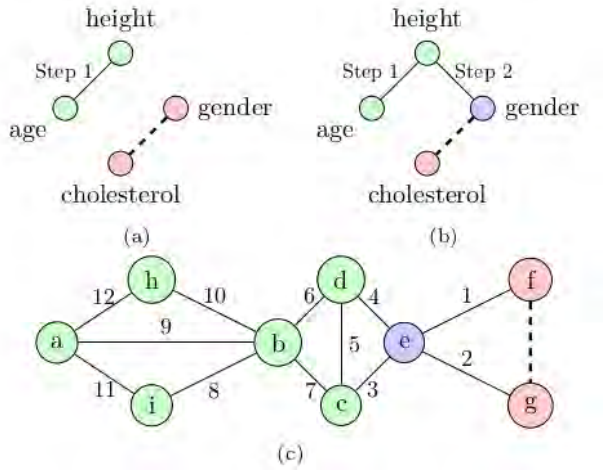


This means that *the vast majority of the computation could be avoided if we knew which edges would lead to the same evaluation of the model*.

This observation is, quite simply, what the technologies that we developed aim at leveraging on: showing how to exactly predict that an edge will need to be re-examined, and designing an algorithm that utilizes that knowledge to learn junction-tree models several orders of magnitude faster than the current state-of-the-art methods.

Our experiments on real-world datasets with up to 2,000 variables show that our algorithm, Prioritized Chordalysis, can search for a junction-tree model about 4 orders of magnitude faster than state-of-the-art techniques, without making any additional assumption.

Let us motivate over a few simple examples why only a very limited number of edges need to be re-examined at each step of the search.



- Case 1: disconnected components. Consider the model of a joint distribution over four variables (age - a, height - h, gender - g and cholesterol - c) illustrated figure (a) on the right. Starting with a model considering that the 4 variables are independent, the first step consists of finding which one of the 6 edges will result in the best model. To this end, the addition of every single edge is evaluated. Let us assume that this model is the one including edge {a,h}, i.e. including the correlation between age and height. The second step is then going to assess the addition of every single edge again. The score[4] associated with the addition of edge {g,c}remains identical, regardless of it being added at the first or second step, because associated variables are not in the same connected components of the graph, and hence not "interacting" in the model; cholesterol and gender are independent of age and height. As a result, this edge need not be evaluated again at the second step.
- Case 2: empty minimal separator. Consider now figure (b) that results from including the interaction {h,g} at step 2. The third step is then going to examine the addition of every remaining edge again. The score associated with the addition of edge {g,c} is identical, regardless of it being added at the first, second or third step, because adding {g,c} will "explain" the same quantity of information in all three models; cholesterol is independent of age and height given gender. We will see that this is due to an empty minimal separator between g and c: $S_{\{gc\}} = \emptyset$, i.e. there is no vertex to remove from the graph to disconnect g from c. As a result, this edge need not be evaluated again at the second and third step.
- Case 3: identical minimal separator. Consider the more elaborate model over 9 variables illustrated in figure (c) above, where the numbers on the edges indicate the steps at which they were added. We show that from step 3, the addition of edge {f,g} to any successively refined model need not be evaluated again and that the score of adding {f,g} will remain invariant. This is motivated by the fact that, from step 3 on-wards, removing the vertex e disconnects f from g ($S_{\{fg\}} = \{e\}$). In consequence, the last time that the addition of this edge needs to be evaluated is at step 3.

We will now prove the validity of these intuitions. It is interesting to observe that being able to tell if an edge has to be re-evaluated is not sufficient, because the

[4] Note that our observations are valid for different scores such as p-value computed from log-likelihood ratios, Kulbach Leibler divergence and MDL/MML scores. In the remainder of this report, we focus on p-value.

search process will still enumerate over all the edges at every step. This enumeration prevents us from scaling to datasets with thousands of variables, because there are $O(M^2)$ M variables. We will show that Prioritized Chordalysis can precisely identify the edges that have to be re-evaluated, and use this information to maintain a data structure that makes it possible avoid such enumeration.

2.1 What edges require re-examination?

We have shown in [12] that computing the statistical significance (p-value) of replacing a current reference model $M^\star$ by a candidate model M^c requires two elements: the difference in the fit G_r^2 and the difference in the complexity df_r. We now develop these elements for our target class of models, i.e. junction tree models.

DEFINITION 1: Let $G = \{V, E\}$ be an undirected graph and two vertices $a, b \in V$. The set of vertices $S \subset V$ is a (a,b)-separator if removing S from G separates the vertices a and b into different connected components. If no proper subset of S is an (a,b)-separator, then S is a minimal (a-b)-separator, noted S_{ab}.

Moreover, we have recently shown that:

THEOREM 1 [12]: If two decomposable models $M^c \subset M^\star$ differ only in one edge (a,b), then:

$$\begin{aligned} \mathrm{G}^2{}_r \;\; = \;\; & 2 \cdot N\big(\mathrm{H}(S_{ab} \cup \{a\}) \; + \mathrm{H}(S_{ab} \cup \{b\}) \\ & - \mathrm{H}\left(S_{ab} \cup \{a, b\}\right) - \mathrm{H}(S_{ab})\big) \end{aligned}$$

where H(.) denotes the entropy.

We can thus formulate the following theorem:

THEOREM 2: Let $M_1^\star$ and $M_2^\star$ be two reference models selected at different steps of the search, $G_1^\star = \{V, E_1^\star\}$ and $G_2^\star = \{V, E_2^\star\}$ their associated graphs, and a, b two verticies such that there is no edge between a and b in either models and $G_1^\star \cup \{a, b\}$ and $G_2^\star \cup \{a, b\}$ are both chordal graphs (i.e. adding {a,b} to either graphs keeps the models in the class of junction-tree models). If S is a minimal (a,b)-separator in $G_1^\star$ and $G_2^\star$ ($S_{ab}^{\star 1} = S_{ab}^{\star 2}$), then the p-value associated with the addition of {a,b} to $M_1^\star$ is identical to the p-value associated with the addition of {a,b} to $M_2^\star$.

Proof. Direct from simplification of G_r^2 with $S_{ab}^{\star 1} = S_{ab}^{\star 2}$. ■

A direct consequence of this theorem is that the p-value associated with the addition of an edge only has to be re-evaluated between two steps of LLA if its minimal separator changes between these steps. The possible gain in computation then depends upon how frequently do minimal separators actually change between successive steps of the search. This obviously depends on the underlying structure of the dataset; we can however bound the maximum number of edges that will change between two steps of the search.

THEOREM 3: The number of edges that need to be re-examined after adding edge (a,b) to the current reference model is at most $2(M-1) - |N(a)| - |N(b)|$, where $N(x)$ designates the neighbours of x, i.e. only $O(M)$ edges require re-examination at every step.

Proof. Adding (a,b) to a chordal graph results in the addition of only one maximal clique: $C_{ab} = S_{ab} \cup a \cup b$ [32, Section 3.2.1]. Any new edge added to the clique-graph of the graph will have C_{ab} as one of its endpoints [32, Theorem 4.3] (note that we use the term "clique-graph" as defined in [31]). It results that any edge impacted by the addition of (a,b) has either the form (a,x) or (b,x) [32, Proof to

Theorem 4.3]. Given that a can at most be connected to $M-1$ vertices and that it is already connected to $|N(a)|$ of them, there are at most $M-1-|N(a)|$ edges of the form (a,x). Similar reasoning for b. ■
This fundamental result establishes that, in the worst-case scenario, only $O(M)$ edges have to be re-examined at each step. This strongly contrasts with the state-of-the-art techniques that require examination of all $O(M^2)$ possible edges.

2.2 How to select all edges that need to be re-examined?
We have shown in the last subsection that an edge needs to be re-examined between two steps of the search if and only if the associated minimal separator has changed between these two steps. The naive way to select all the edges that need to be re-examined at every new step would then be to iterate over all edges (a,b) and select those for which the minimal separator has changed. However, we have seen that iterating over all possible edges at every step of the search is precisely the limiting factor to scale up to datasets with thousands of variables. Furthermore, even this naive selection would require prohibitive calculations, because finding all S_{ab} itself requires $O(|V|+|E|)$ operations for chordal graphs [33]. In this subsection, we show how both these problems can be solved using an advanced graph-theoretical data structure - the clique-graph [31]:

1. We demonstrate that, for all edges (a,b)$ that are considered for addition to successive reference models $M^{\star}$, their minimal (a,b)-separators can be efficiently derived from the clique-graph.
2. We take an existing algorithm that aims at maintaining the clique-graph data structure when iteratively adding edges to the supporting graph [32], and show how to modify it to keep track of all minimal (a,b)-separators.

2.2.1. Minimal vertex separators align with edges of the clique-graph
The clique-graph structure is an ideal base-structure for our task of keeping track of all minimal separators between the vertices.
DEFINITION 2 [31, Definition 2]: Let G be a chordal graph. The clique-graph $C(G)=\{V_c,E_c\}$ is defined by:

- V_c is the set of maximal cliques of G
- (C_1,C_2) belongs to E_c iff $C_1\cap C_2$ is a minimal (a,b)-separator for each $a\in C_1\backslash C_2$ and each $b\in C_2\backslash C_1$

We now formulate the theorem that is the base for tracking the minimal separators.
THEOREM 4: If (a,b) can be added to a chordal graph G while maintaining its chordality, then $S_{ab}=C_a\cap C_b$ where $(C_a,C_b)\in E_c, a\in C_a$ and $b\in C_b$.
Proof. If adding (a,b) maintains the chordality of G then $\exists(C_a,C_b)$ in $C(G)$ such as $a\in C_a$ and $b\in C_b$ [32, Lemma 3.1]. By Definition 2, if $(C_a,C_b)\in E_c$, then $C_a\cap C_b$ is a minimal (a,b)-separator. ■

2.2.2. Minimal vertex separators align with edges of the clique-graph
We have demonstrated in Theorem 4 that for all edges (a,b), the minimal (a,b)-separator S_{ab} can be obtained from edges (C_a,C_b) of the clique-graph, such as $a\in C_a$ and $b\in C_b$. A naive way of keeping track of all the minimal separators could thus be to iterate over the edges $C_1,C_2)$ of the clique-graph, and for each one of them, to iterate over all pairs of vertices $(x,y), x\in C_1\backslash C_1\cap C_2, y\in C_2\backslash C_2\cap C_1$

and memorize that $S_{xy} = C_1 \cap C_2$. This would however lead to a vast amount of unnecessary computations, because most of the structure of the clique-graph remains unchanged when adding an edge to the associated (normal) graph.

We refine here the state-of-the-art algorithm for the iterative update of clique-graphs [32] in order to keep track of all minimal vertex separators. Note that we only detail the modified parts of [32]'s algorithm.[5] The theoretical contribution of this part of the paper concerns ε_M^f - a boolean MxM-matrix informing about the eligibility of any edge for addition to the current graph - and its iterative update:

1. We make ε_M^f a function that associates to any pair of vertices (a,b) its eligibility, its minimal separator S_{ab} and the clique-graph edge $(C_a, C_b) \in E_c$ such that $a \in C_a$, $b \in C_b$ and $C_a \cap C_b = S_{ab}$, i.e. the two nodes of the clique-graph allowing a to be connected to b in G.
2. Following our Theorem 4, every time a new edge (C', C_{ab}) is added to the clique-graph as a result of adding (a,b) to the graph, we set $\varepsilon_M^f(x,a)$ to $(\text{true}, C' \cap C_{ab}, C', C_{ab})$ for all (x,a) such as $x \in C' \backslash C_{ab}$, $a \in C_{ab} \backslash C'$. Similarly for b.
3. Every time an edge (C_1, C_2) is deleted from $C(G)$ as a result of adding (a,b) to G and noting that such (C_1, C_2) will follow $C_1 \cap C_2 = S_{ab}$, we set $\varepsilon_M^f(x,a)$ to $(\text{false}, \_, \_, \_)$ for all pairs (x,y) such that x (resp. y) is in the same connected component as a (resp. b) in $G - S_{ab}$.

In addition, note that the scientific community has challenged the correctness of [32]'s algorithm, in particular for the case where G is made of several connected components [34] which leads to empty minimal separators. We attribute this to a few unfortunate typos present in [32], to the use of an imprecise vocabulary,[6] and to the absence of any available implementation of the algorithm. We have clarified, corrected and extended [32]'s algorithm. Our algorithm can easily been reversed back to the original algorithm by only considering the boolean values in $\varepsilon_M^f(x,a)$. Note that the validity of our implementation has been carefully checked and tested over hundreds of experiments, where we verified that it led to the same results as algorithms which do not make use of the clique-graph [35].

2.3 Efficiently iterating over the best edges

This subsection describes the last component of our algorithm: how to prevent enumeration over all possible edges at every step.

At every step, the standard model-search frameworks consider all the possible modifications of the current reference model. This requires iteration over all $O(M^2)$ possible edges, which is the limiting factor to perform the search for datasets with thousands of variables. As there are at most $\binom{M}{2}$ steps, state-of-the-art algorithms can *all* lead to the examination of $O(M^4)$ edges.

---

[5] The reader can refer to the original paper and to our implementation available at http://github.com/fpetitjean/chordalysis for more details

[6] An example is the use of ``connected'' which can be interpreted as the presence of a direct edge between two vertices, or as the existence of a path connecting these vertices.

Our Prioritized-Chordalysis approach uses a priority queue to store the edges that have to be successively considered for addition to the current reference model. We keep the edges ordered by their associated statistical significance. As we have seen in Section 2.1, if the minimal separator associated with an edge does not change from one step to another, neither does the statistical significance associated with this edge. This means that, at every step, the only edges that are going to change in the queue, are 1) the edges that are not eligible anymore because they would not keep the graph chordal, 2) the edges that are newly eligible and 3) the edges that have had a change of minimal separator.
We have shown in Section 2.2 that such changes are all associated with the addition and deletion of edges in the clique-graph: adding a clique-graph edge enables new edges (or change their minimal separator) while removing a clique-graph edge disables edges. To keep the explanation simple, and because we will see that this does not change the overall complexity, we consider a priority queue based on a heap data structure, with retrieval and removal of the minimum in $O(1)$, and insertion/deletion of an element in $O(\log n)$. We now prove that, even in the worst case, our solution exhibits a far better complexity than state-of-the-art methods.

Initialization. At the start, all pairs of edges are sorted and added to the queue, which requires $O(M^2 \log(M))$ operations.
Edge addition. Any new clique-graph edge has C_{ab} as its endpoint (see proof to Theorem 3). In consequence, any edge impacted by the addition of (a,b) has either the form (x,a) or (x,b). As a (and b) cannot be connected to more than M-1 vertices, at every step of the search, at most $2(M-1)$ edges might be added to the queue; resulting in a quasi-linear complexity with the number of variables for each of the $O(M^2)$ possible steps, thus $O(M^3 \log M)$.
Edge deletion. Any edge that is removed from the priority queue has obviously to have been added to it. As there are at most $O(M^2 \log M + M^3 \log M)$ such additions, there will also be at most $O(M^3 \log M)$ such deletions.
Overall. For k steps performed, our algorithm thus requires only $O(kM \log M)$ operations; every step of LLA exhibits a quasi-linear complexity with the number of variables. This starkly contrasts with the quadratic $O(kM^2)$ complexity of state-of-the-art algorithms [32,34,12,13]. Our experiments will show that this difference makes it possible to gain efficiency by several orders of magnitude and allows us to perform the search for a statistically significant junction-tree model for datasets with thousands of variables.

3. Experiment:

We have shown in Section 2 that Prioritized Chordalyis dominates the state of the art in terms of algorithmic complexity. This section seeks to demonstrate its computational superiority on real-world datasets. Note that this section does not seek to further assess the relevance of χ^2 goodness-of-fit tests for learning graphical models, because it has long been accepted by the community. Rather, our experiments seek to demonstrate that we can achieve further scalability without sacrificing the statistical soundness of the scoring methods. To this end, we consider four successively refined algorithms for searching junction-tree models, starting from the current state of the art for high-dimensional data [12] and progressively incorporating the contributions of this paper:

Version 1: We start with Chordalysis: the first method that can perform the search on high-dimensional data [12].
Version 2: We integrate the clique-graph update algorithm from [32] into Version 1.
Version 3: We add to **Version 2** the ability to keep track of the minimal (a,b)-separators.
Version 4 -- Prioritized Chordalysis: We add to **Version 3** the ability to keep track of the best edges to be successively added in a priority queue.

As we have demonstrated in Section 2, the worst-case complexity only depends on the number of variables. This is a consequence of the number of edges depending on the number of vertices. However, the number of edges to be discovered from data depends on the actual dependencies that can be found in data. If the data is drawn from a probability distribution where all variables are actually independent, then the process will quickly finish. In contrast, real-world datasets often exhibit numerous high-order correlations, leading to more computation time. In addition, the quantity of data has also a significant impact on the computation time. This can seem counter-intuitive because the scoring of an edge depends on four entropies only, and each entropy can be naively computed with a quasi-linear complexity with the size of the dataset. However, increased data quantity allows more edges to be identified as statistically significant and will thus often lead to a very significant increase in the computation time. This is well exemplified by the toss of a coin and the associated decision: if we toss the coin 100 times and we observe 51 heads and 49 tails, we cannot state that the coin is unbalanced. However, it we toss it 100,000 times and 51,000 heads and 49,000 tails, and while this is the same proportion of heads/tails, statistical tests tell us that we can confidently state that it is very unlikely that the coin is balanced. This phenomenon is similar to the one observed with the learning of decision trees: larger quantities of data will tend to create deeper trees. This is why we use a broad range of real-world datasets, with both various number of variables and various quantities of data:

- Mushroom: the classical mushroom dataset, 22 variables, 8k examples [36].
- EPESE: epidemiological study of the elderly, 25 variables, 14k examples [37].
- Internet: demographic information on internet users, 70 variables, 10k examples [36].
- CoIL2000: insurance customer management, 86 variables, 6k examples [38].
- MITFace: face recognition dataset, discretized to 4 bins using equal frequency, 362 variables, 31k examples [39].
- Finance: stock performance of the companies listed in the S&P500 over 20 years of trading, 500 variables.
- Protein: Multiple alignment of the Serpin family of proteins, 750 variables, 212 proteins [42].
- Orphamine: Frequency of occurrence of 1,260 symptoms for 2,600 rare diseases, 1,260 variables, 2,600 examples [40].
- ABC: Use of the 500 most interesting words in all the news articles about Melbourne published by the Australian Broadcasting Network (ABC), 500 variables, 35k examples.
- NYT: Use of the 2,000 most interesting words in 10% of the articles published by the New York Times from 1987 to 2007, 2,000 variables, 180k examples [41].

Where licensing restrictions permit us to do so, we have made these datasets available at http://bit.ly/PrioChordalysisRes.

Figure 3 presents the computation time required to perform LLA for every version of the algorithm on these real-world datasets. Note that the graphs associated with each dataset are provided at http://bit.ly/PrioChordalysisRes. These results confirm the superiority of our method. Prioritized Chordalysis is the fastest method for all datasets. Moreover, for all datasets with more than 100 variables (from MIT Face), Prioritized Chordalysis performs the search with about 4 orders of magnitude faster than the state of the art. For example, for the ABC dataset – which comprises 500 variables – Prioritized Chordalysis performs the search in 27 seconds while Chordalysis (Version 1) requires 39 hours (to obtain exactly the same result); this is more than a 5,200x speedup.

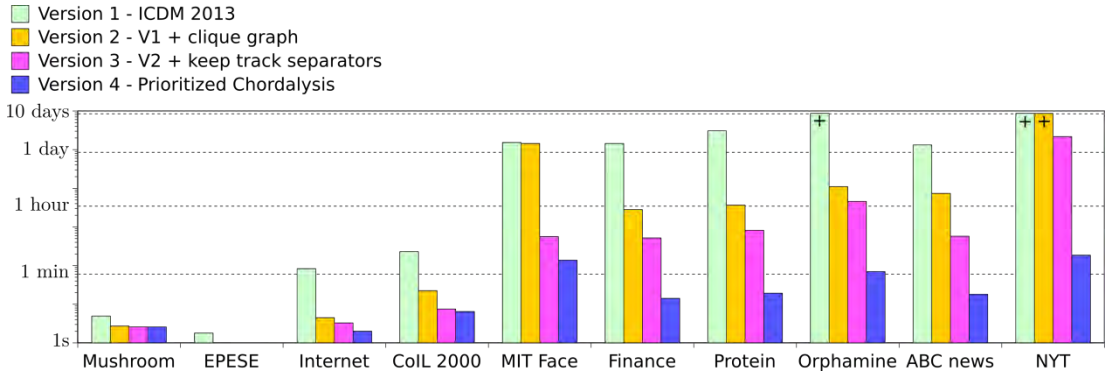


Figure 3: Comparison of the computation time required to perform a forward search on various real-world datasets. "+" indicates that the computation did not finish within 10 days of computation.

This is a major result that makes it possible to tackle datasets with *thousands* of variables. For such datasets, our experiments indeed show that Prioritized Chordalysis makes it possible to perform the search in seconds or minutes, when the state of the art requires days. For example, for the `NYT` dataset – which comprises 2,000 variables – Prioritized Chordalysis performs the search in only 3 minutes while Version 1 could not provide any result in 10 days of computation.

Furthermore, we can observe that all the successive elements that we have introduced in this paper play a major role in making the search scalable to very high-dimensional datasets. Each of the contributions that we have made in this paper - from providing a complete and correct clique-graph-update algorithm, to keeping track of the minimal separators in order to maintain the possible modifications in a priority queue - gains one to two orders of magnitude, depending on the dimensionality of the dataset, amount of available evidence, and complexity of the underlying joint distribution.

Finally, we examine the scalability of Prioritized Chordalysis, on a dataset with increasing number of variables. The `NYT` dataset is a good test bed for this task because 1) it is our biggest dataset with 180,000 instances and 2,000 variables and 2) its variables are ordered (occurrence frequency of every word), which makes its study possible with an increasing number of variables; the most frequent words first.

Figure 4 presents the results of this experiment. We can observe that our proposed algorithm, Prioritized Chordalysis, greatly dominates all other methods. Moreover, we can see that the magnitude of the improvement of Prioritized Chordalysis actually increases over time, i.e. the plots get farther apart as the number of variables increases.

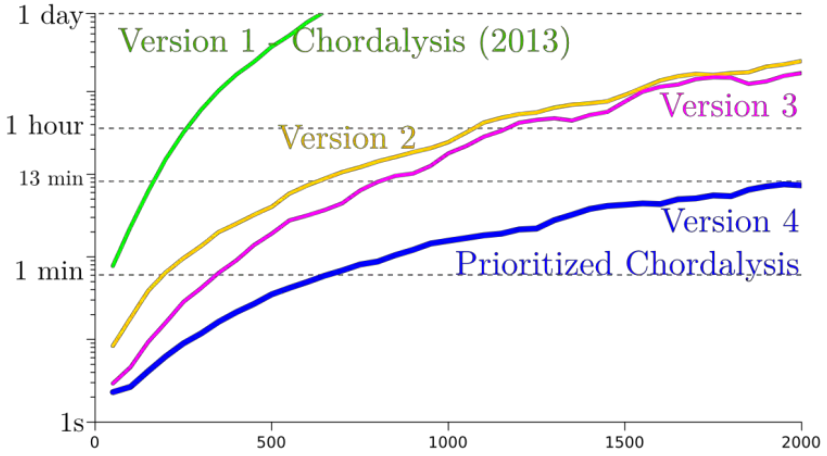


Figure 4: Comparison of the computation time required to perform the search with regard to the number of variables used –dataset `NYT`. We limited the discovery to the first 100 edges to limit the computation time.

Interestingly, we can also see that when the number of variables increases, Version 2 tends to perform as fast as Version 3. This is because the time required to find the minimal separator of every edge from the clique graph (Version 2) becomes negligible relative to maintaining the structure of the clique graph. In consequence, tracking the minimal separators (Version 3) tend to provide only marginal improvement over Version 2. Note however that Version 4 (Prioritized Chordalysis) keeps track of the minimal separators to maintain the priority queue; this element is thus necessary to obtain the exhibited improvement.

4. Results and Discussion:

Being able to predict any variable from all the available information about a system – Omnidirectional learning - is critical for many applications. In this project, we showed how junction trees are an excellent class of models to perform this task, because they have exact and fast marginalization algorithms available. The remaining scientific lock was the scalability of learning such models.

With the contributions of this project, we have showed how such models can be learned for a large class of real-world data assets, and this on a standard desktop computer only.

More specifically, we made the following contributions:

1. We proved that only a very small subset of edges has to be considered at each step of the search.
2. We demonstrated how to efficiently find this subset of edges.
3. We showed how to efficiently keep track of the best edges to be subsequently added to the initial model.

Our experiments, carried out on real datasets with up to 2,000 variables, showed that our contributions make it possible to gain about 4 orders of magnitude in

computation time, making it possible to learn effective junction-tree models of joint distributions in seconds instead of days. Because efficient marginalization exists for these models, our contributions make omnidirectional learning feasible on any standard desktop computer for virtually all datasets with up to thousands of variables.

The research was published and presented at the SIAM International Conference on Data Mining and received **Best Paper Honorable mention.**

To ensure broad use and uptake of the outcomes of this research project, we have released Prioritized Chordalysis open-source on Github at http://github.com/fpetitjean/chordalysis. This will allow industry and many fields of science to unlock additional value from their new and existing data assets.

5. References

[1] Aggarwal, Charu C., ed. "Data Classification: Algorithms and Applications". CRC Press, 2014.

[2] Saar-Tsechansky, M. and Provost, F. "Handling missing values when applying classification models", *Journal of Machine Learning Research*, Vol. 8, pp. 1625-1657, 2007.

[3] J. H. Friedman, R. Kohavi, and Y. Yun. "Lazy decision trees". *Proceedings of the Thirteenth National Conference on Artificial Intelligence and the Eighth Innovative Applications of Artificial Intelligence Conference*, pp. 717-724, 1996.

[4] D. Koller and N. Friedman, "Probabilistic Graphical Models, Principles and Techniques", The MIT Press, 2009.

[5] R. Christensen. "Log-linear models and logistic regression - Model Selection Methods and Model Evaluation". *Springer Texts in Statistics*, 1997, pp. 211-257.

[6] Mooij, J; Kappen, H (2007). "Sufficient Conditions for Convergence of the Sum–Product Algorithm". *IEEE Transactions on Information Theory* 53 (12): 4422–4437.

[7] G. Cooper and E. Herskovits. "A Bayesian method for the induction of probabilistic networks from data" *Machine learning,* 9, 1992.

[8] W. Buntine. "Theory refinement on Bayesian networks", *Uncertainty in Artificial Intelligence*, 1991.

[9] L. Su-In, V. Ganapathi, and D. Koller. "Efficient Structure Learning of Markov Networks using l_1-Regularization". *Advances in Neural Information Processing Systems,* 2006.

[10] V. Gogate, W.A. Webb, and P. Domingos, "Learning Efficient Markov Networks", in *Advances in Neural Information Processing Systems*, 2010.

[11] D. Koller, N. Friedman, L.Getoor, and B.Taskar. 'Graphical Models in a Nutshel', *Introduction to Statistical Relational Learning*, 2007.

[12] F. Petitjean, G. I. Webb and A. E. Nicholson, "Scaling log-linear analysis to high-dimensional data" in *IEEE International Conference on Data Mining*, 2013.

[13] F. Petitjean, L. Allison, G. Webb. "A statistically efficient and scalable method for log-linear analysis of high-dimensional data" in *IEEE International Conference on Data Mining*, *to appear,* 2014.

[14] Cortes, C. and Vapnik, V. "Support-vector networks". *Machine Learning* 20 (3): 273, 1995.

[15] Breiman, Leo, "Random Forests". *Machine Learning* 45 (1): 5–32, 2001.

[16] Shelby J Haberman, "The analysis of frequency data", *University of Chicago Press*, 1974.

[17] J. Ganz and D. Reinsel, "The Digital Universe Study," *International Data Corporation*, Framingham, 2012.

[18] R. Bekkerman, M. Bilenko and J. Langford, "Scaling up machine learning: Parallel and distributed approaches". *Cambridge University Press*, 2011.

[19] E. Keogh and A. Mueen. "Curse of Dimensionality". *Encyclopedia of Machine Learning*. pp 257-258. 2010.

[20] J. S. Vitter, "External Memory Algorithms and Data Structures: Dealing with MASSIVE DATA," *ACM Computing Surveys,* 33, 2001.

[21] O. Maron and A. Moore, "The Racing Algorithm," *Artificial Intelligence Review,* vol. 11, pp. 193-225, 1997.

[22] J. S. Simonoff, "Smoothing Methods in Statistics," *Springer Series in Statistics*, 1996.
[23] Y.W. Teh. Coling, "A Hierarchical Bayesian Language Model based on Pitman-Yor Processes," *Association for Computational Linguistics,* 2006.
[24] W.L. Buntine and S. Mishra, "Experiments with non-parametric topic models", *ACM International Conference on Knowledge Discovery and Data Mining*, 2014.
[25] M. J. Wainwright, P. Ravikumar and J. D. Lafferty, "High-Dimensional graphical Model Selection Using l_1-Regularized Logistic Regression," in *Conference on Neural Information Processing Systems (NIPS)*, 2006.
[26] S. Peter, C. Glymour, and R. Scheines. "Causation, prediction, and search," Vol. 81. *The MIT Press*, 2000.
[27] A. Deshpande, M. N. Garofalakis and M. I. Jordan, "Efficient Stepwise Selection in Decomposable Models," *International Conference on Uncertainty in Artificial Intelligence,* 2001.
[28] Lafferty, J., McCallum, A., Pereira, F.. "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *International Conference on Machine Learning*, 2001.
[29] Pernkopf, F. and Bilmes, J. "Discriminative versus Generative Parameter and Structure learning of Bayesian Network Classifiers", *International Conference on Machine Learning,* 2005.
[30] Krähenbühl, P. and Koltun, V., "Parameter Learning and Convergent Inference for Dense Random Fields", *International Conference on Machine Learning, 2013.*
[31] P. Galinier, M. Habib, and C. Paul, "Chordal graphs and their clique graphs," in *Graph-Theoretic Concepts in Computer Science*, Lecture Notes in Computer Science, Springer, 1995, pp. 358–371.
[32] A. Deshpande, M. Garofalakis, and M.I. Jordan, "Efficient stepwise selection in decomposable models," in *Uncertainty in Artificial Inteligence*, 2001, pp. 128–135.
[33] P.S. Kumar and C.E.V. Madhavan, Minimal vertex separators of chordal graphs, Discrete Applied Mathematics, 89 (1998), pp. 155–168.
[34] S. Altmueller and R.M. Haralick, Practical aspects of efficient forward selection in decomposable graphical models, in *IEEE Int. Conf. on Tools with Artificial Intelligence*, 2004, pp. 710–715.
[35] A. Berry and R. Pogorelcnik, A simple algorithm to generate the minimal separators and the maximal cliques of a chordal graph, *Information Processing Letters*, 111 (2011), pp. 508–511.
[36] S. Hettich and S.D. Bay, UCI KDD archive, 1999.
[37] J.O. Taylor, R.B. Wallace, A.M. Ostfeld, and D.G. Blazer, Established Populations for Epidemiologic Studies of the Elderly, 1981-1993. http://dx.doi.org/10.3886/ICPSR09915, 1998.
[38] P. van der Putten and M. van Someren, A Bias-Variance Analysis of a Real World Learning Problem: The CoIL Challenge 2000, Machine Learning, 57 (2004), pp. 177–195.
[39] MIT Center For Biological and Computation Learning, CBCL Face Database #1. http://www.ai.mit.edu/projects/cbcl, 2000.
[40] Orphanet, An online database of rare diseases and orphan drugs. http://www.orpha.net, 2014.
[41] Sandhaus E., The New York Times Corpus. https://catalog.ldc.upenn.edu/LDC2008T19, 2008.
[42] J.A. Irving, R.N. Pike, A.M. Lesk, and J.C. Whisstock, Phylogeny of the serpin superfamily: Implications of patterns of amino acid conservation for structure and function, Genome Research, 10 (2000), pp. 1845–1864.

List of Publications and Significant Collaborations that resulted from your AOARD supported project: In standard format showing authors, title, journal, issue, pages, and date, for each category list the following:

a) papers published in peer-reviewed journals,

b) papers published in peer-reviewed conference proceedings,

- F. Petitjean and G.I. Webb. Scaling log-linear analysis to datasets with thousands of variables. *SIAM Conference on Data Mining*, pp. 469-477, 2015. Best Paper Award Honorable Mention.

c) papers published in non-peer-reviewed journals and conference proceedings,

d) conference presentations without papers,

e) manuscripts submitted but not yet published:

f) provide a list any interactions with industry or with Air Force Research Laboratory scientists or significant collaborations that resulted from this work.

Attachments: Publications a), b) and c) listed above if possible.